C BYREGOWDA INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Affiliated to Visvesvaraya Technological University "JnanaSangama",

Belgaum–560018.

Laboratory Manual

COMPUTER NETWORK LABORATORY (BCS502)

V Semester (CBCS Scheme)

Prepared by

Prof. SUMARANI H

Assistant Professor

Prof. SWATHI J K

Assistant Professor

Prof. ANITHA P

Assistant Professor

Scrutinized by
Dr.VASUDEVA.R
Associate Professor, HOD

C. BYREGOWDA INSTITUTE OF TECHNOLOGY

(Approved by AICTE New Delhi, Accredited by NAAC with B++ Grade, Recognized by Govt. of Karnataka, Affiliated to VTU Belagavi, ISO 9001:2015Certified Institute)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Kolar – Srinivaspur Road, Kolar – 5631012024-25

COMPUTER NETWORK LABORATORY

[As per Choice Based Credit System (CBCS) scheme]
(Effective from the academic year 2018)
SEMESTER – V

Subject Code BCS502 **IA Marks 20**
Number of Lecture Hours/Week 01I + 02P **Exam Marks 80**
Total Number of Lecture Hours 40 **Exam Hours 03**

CREDITS – 02

Course objectives: This course will enable students to
· Demonstrate operation of network and its management commands
· Simulate and demonstrate the performance of GSM and CDMA
· Implement data link layer and transport layer protocols.

Description (If any):
For the experiments below modify the topology and parameters set for the experiment and take multiple rounds of reading and analyze the results available in log files. Plot necessary graphs and conclude using any **suitable** tool.

Lab Experiments:

PRACTICAL COMPONENT OF IPCC

Sl.NO	Experiments
1	Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth, and find the number of packets dropped.
2	Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.
3	Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.
4	Develop a program for error detecting code using CRC-CCITT (16- bits).
5	Develop a program to implement a sliding window protocol in the data link layer.
6	Develop a program to find the shortest path between vertices using the Bellman-Ford and path vector routing algorithm.
7	Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present.
8	Develop a program on a datagram socket for client/server to display the messages on client side, typed at the server side.
9	Develop a program for a simple RSA algorithm to encrypt and decrypt the data.
10	Develop a program for congestion control using a leaky bucket algorithm.

Study Experiment / Project: NIL

Course outcomes (Course Skill Set):
At the end of the course, the student will be able to:

- Explain the fundamentals of computer networks.
- Apply the concepts of computer networks to demonstrate the working of various layers and Protocols in communication network.
- Analyze the principles of protocol layering in modern communication systems.
- Demonstrate various Routing protocols and their services using tools such as Cisco packet Tracer

Note: For the Simulation experiments modify the topology and parameters set for the experiment and take multiple rounds of reading and analyze the results available in log files. Plot necessary graphs and conclude using NS2 or NS3. Installation procedure of the required software must be demonstrated, carried out in groups, and documented in the report. Non simulation programs can be implemented using Java.

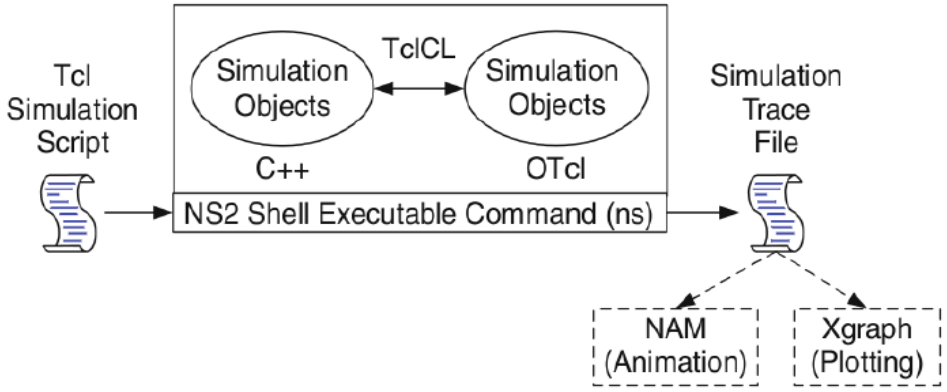
Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIEmade zero.

Introduction to NS-2:

- Widely known as NS2, is simply an event driven simulation tool.
- Useful in studying the dynamic nature of communication networks
- Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2.
- In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviors.

Basic Architecture of NS2



Tcl scripting

• Tcl is a general purpose scripting language. [Interpreter]

• Tcl runs on most of the platforms such as Unix, Windows, and Mac.

• The strength of Tcl is its simplicity.

• It is not necessary to declare a data type for variable prior to the usage.

Basics of TCL

Syntax: command arg1 arg2 arg3

- **Hello World:**

puts stdout{Hello, World!}
Hello, World!

- **Variables:** Command Substitution

set a 5 set len [string length foobar]
set b $a set len [expr [string length foobar] + 9]

- **Simple Arithmetic:** expr 7.2 / 4

- **Procedures**

```
proc Diag {a b} {
set c [expr sqrt($a * $a + $b * $b)]
return $c }
puts —Diagonal of a 3, 4 right triangle is [Diag 3 4]‖
```

Output: Diagonal of a 3, 4 right triangle is 5.0

- **Loops**

```
while{$i < $n} { for {set i 0} {$i < $n} {incr i} {
. . . . . .
} }
```

Initialization and Termination of TCL Script in NS-2 :

An ns simulation starts with the command

set ns [new Simulator]

Which is thus the first line in the tcl script? This line declares a new variable as using the set command, you can call this variable as you wish, In general people declares it as ns because it is an instance of the Simulator class, so an object the code[new Simulator] is indeed the installation of the class Simulator using the reserved word new. In order to have output files with data on the simulation (trace files) or files used for visualization (nam files), we need to create the files using —open command:

```
#Open the Trace file
set tracefile1 [open out.tr w]
$ns trace-all $tracefile1
#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile
```

The above creates a dta trace file called **—out.tr** and a nam visualization trace file called **—out.nam**.
Within the tcl script, these files are not called explicitly by their names, but instead by pointers that are declared above and called —tracefile1 and —namfile respectively. Remark that they begins with a # symbol.The second line open the file —out.tr‖ to be used for writing, declared with the letter —w‖.The third line uses a simulator method called trace-all that have as parameter the name of the file where the traces will go.
The last line tells the simulator to record all simulation traces in NAM input format.It also gives the file name that the trace will be written to later by the command $ns flush-trace.In our case,this will be the file pointed at by the pointer —$namfile‖,i.e the file —out.tr‖.
The termination of the program is done using a —finish procedure.

```
#Define a "finish" procedure
Proc finish { } {
global ns tracefile1 namfile
$ns flush-trace
```

```
Close $tracefile1
Close $namfile
Exec nam out.nam &
Exit 0
}
```

The word proc declares a procedure in this case called **finish** and without arguments. The word **global** is used to tell that we are using variables declared outside the procedure. The simulator method **—flush-trace"** will dump the traces on the respective files. The tcl command **—close"** closes the trace files defined before and **exec** executes the nam program for visualization. The command **exit** will ends the application and return the number 0 as status to the system. Zero is the default for a clean exit. Other values can be used to say that is a exit because something fails.

At the end of ns program we should call the procedure —finish and specify at what time the termination should occur. For example, **$ns at 125.0 "finish"** will be used to call **—finish**‖ at time 125sec.Indeed,the **at** method of the simulator allows us to schedule events explicitly.

The simulation can then begin using the command: **$ns run**

Structure of Trace Files:

When tracing into an output ASCII file, the trace is organized in 12 fields as follows in fig shown below, The meaning of the fields are:

Event	Time	From Node	To Node	PKT Type	PKT Size	Flags	Fid	Src Addr	Dest Addr	Seq Num	Pkt id

1. The first field is the event type. It is given by one of four possible symbols r, +, -, d which correspond respectively to receive (at the output of the link), enqueued, dequeued and dropped.
2. The second field gives the time at which the event occurs.
3. Gives the input node of the link at which the event occurs.
4. Gives the output node of the link at which the event occurs.
5. Gives the packet type (eg CBR or TCP)
6. Gives the packet size
7. Some flags
8. This is the flow id (fid) of IPv6 that a user can set for each flow at the input OTcl script one can further use this field for analysis purposes; it is also used when specifying stream color for the NAM display.
9. This is the source address given in the form of —node.port‖.
10. This is the destination address, given in the same form.
11. This is the network layer protocol's packet sequence number. Even though UDP implementations in a real network do not use sequence number, ns keeps track of UDP packet sequence number for analysis purposes
12. The last field shows the Unique id of the packet.

Awk- An Advanced

awk is a programmable, pattern-matching, and processing tool available in UNIX. It works equally well with text and numbers. awk is not just a command, but a programming language too. In other words, awk utility is a pattern scanning and processing language. It searches one or more files to see if they contain lines that match specified patterns and then perform associated actions, such as writing the line to the standard output or incrementing a counter each time it finds a match.

Syntax:

awk option 'selection criteria {action}' file(s)

PROGRAM 1

Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth, and find the number of packets dropped.

```
#Create Simulator
set ns [new Simulator]

#Open Trace file and NAM file
set ntrace [open prog1.tr w]
$ns trace-all $ntrace
set namfile [open prog1.nam w]
$ns namtrace-all $namfile


#Finish Procedure
 proc Finish {}  {
global ns ntrace namfile

#Dump all the trace data and close the files
$ns flush-trace
close $ntrace
close $namfile

#Execute the nam animation file
exec nam prog1.nam &

#Show the number of packets dropped

exec echo "The number of packet drops is " &
exec grep -c "^d" prog1.tr &
exit 0
}

#Create 3 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]

#Label the nodes
$n0 label "TCP Source"
$n2 label "Sink"
#Set the color
$ns color 1 blue

#Create Links between nodes
```

```
#You need to modify the bandwidth to observe the variation in packet drop

$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail

#Make the Link Orientation
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right

#Set Queue Size
#You can modify the queue length as well to observe the variation in packet drop

$ns queue-limit $n0 $n1 10
$ns queue-limit $n1 $n2 10

#Set up a Transport layer connection.

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n2 $sink0
$ns connect $tcp0 $sink0

#Set up an Application layer Traffic

set cbr0 [new Application/Traffic/CBR]
$cbr0 set type_ CBR
$cbr0 set packetSize_ 100
$cbr0 set rate_ 1Mb
$cbr0 set random_ false
$cbr0 attach-agent $tcp0

$tcp0 set class_ 1

#Schedule Events
$ns at 0.0 "$cbr0 start"
$ns at 5.0 "Finish"

#Run the Simulation
$ns run
```

Steps for execution:

1.Open vi editor and type program.

2.Program name should have the extension " .tcl " **vi lab1.tcl**

Save the program by pressing **ESC :wq** and press enter

3.Open vi editor and type awk program. Program name should have the extension ".awk " **vi lab1.awk**

Save the program by pressing **ESC:wq** and press Enter key.

4.Run the simulation program : **ns lab1.tcl** Here "ns" indicates network simulator. We get the topology shown in the snapshot.

5. Now press the play button in the simulation window and the simulation will begins.

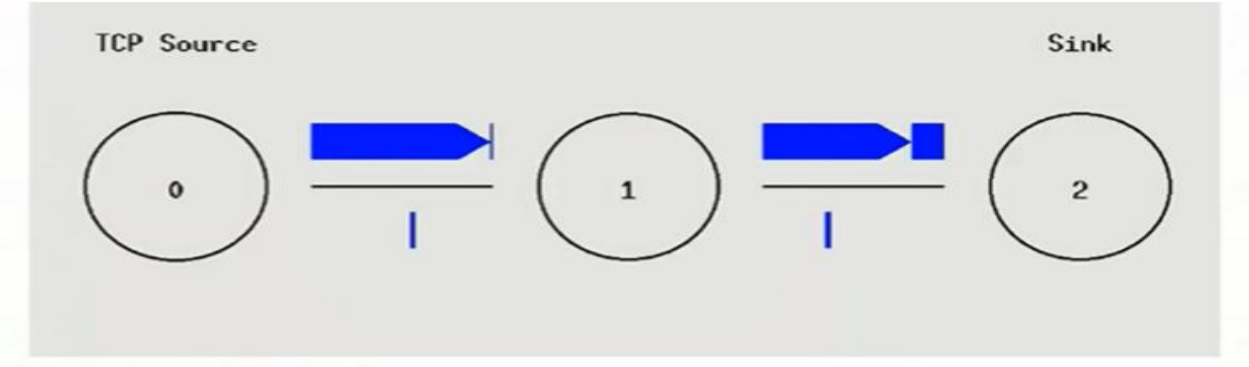
6.After simulation is completed run awk file to see the output:

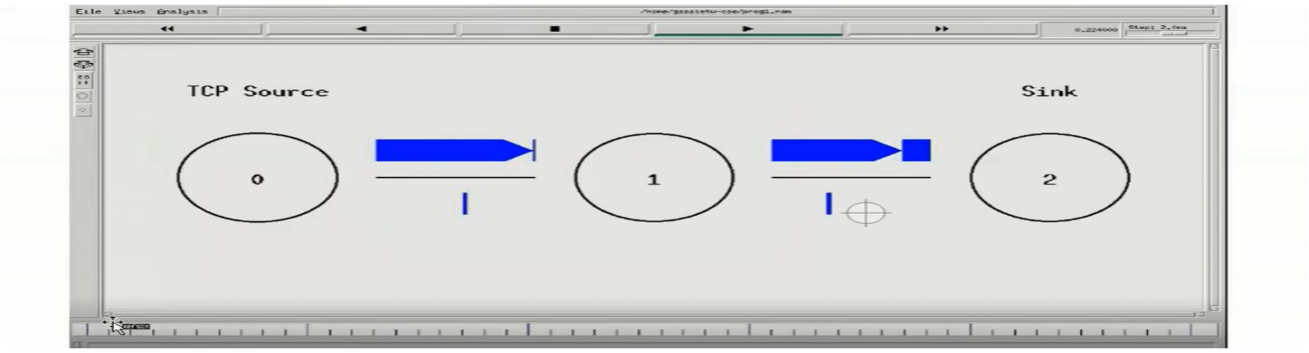
awk –f lab1.awk lab1.tr

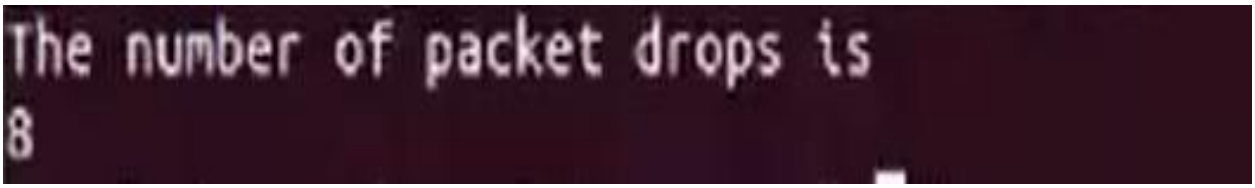
7. To see the trace file contents open the file as : **vi lab1.tr**

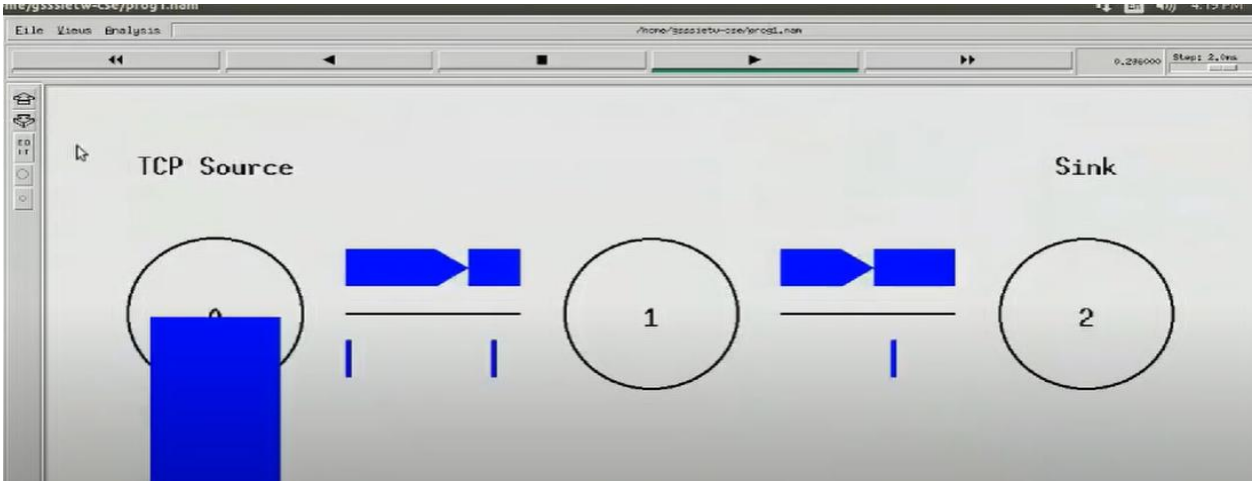
Output:

1 Scenario: No Of packets Dropped

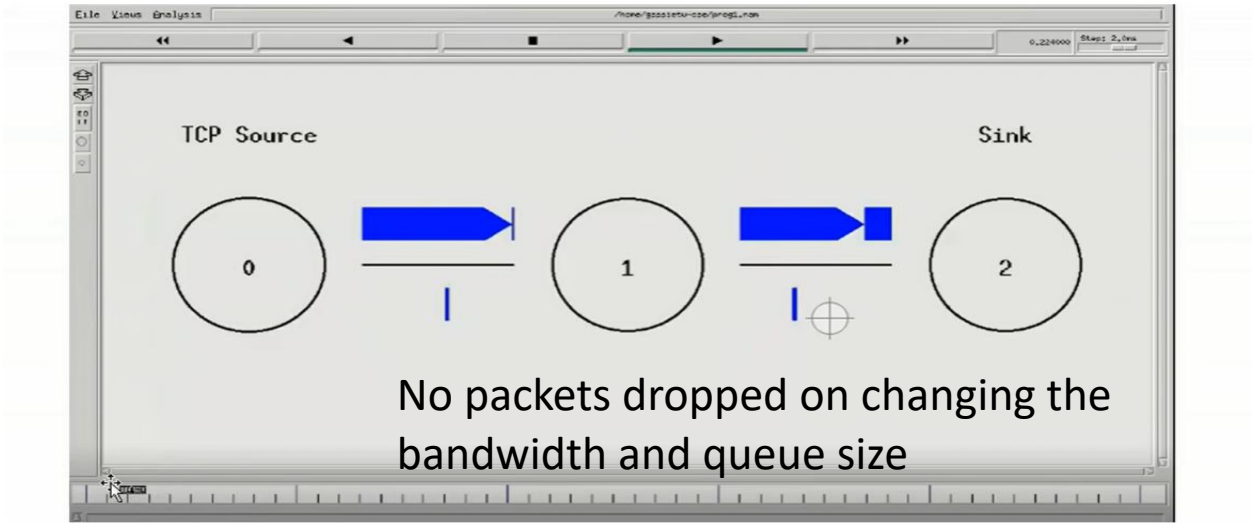








2 Scenario: On changing the Queue size and Bandwidth



PROGRAM 2

Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.

```
#Create Simulator

set ns [new Simulator]

#Use colors to differentiate the traffic
$ns color 1 Blue
$ns color 2 Red

#Open trace and NAM trace file
set ntrace [open prog2.tr w]
$ns trace-all $ntrace
set namfile [open prog2.nam w]
$ns namtrace-all $namfile

#Finish Procedure
proc Finish {} {
global ns ntrace namfile

#Dump all trace data and close the file
$ns flush-trace
close $ntrace
close $namfile

#Execute the nam animation file
exec nam prog2.nam &

#Find the number of ping packets dropped
puts "The number of ping packets dropped are "
exec grep "^d" prog2.tr | cut -d " " -f 5 | grep -c "ping" &
 exit 0
}
#Create six nodes
for {set i 0} {$i < 6} {incr i} {
set n($i) [$ns node]
}

#Connect the nodes
for {set j 0} {$j < 5} {incr j} {
 $ns duplex-link $n($j) $n([expr ($j+1)]) 0.1Mb 10ms DropTail
}
```

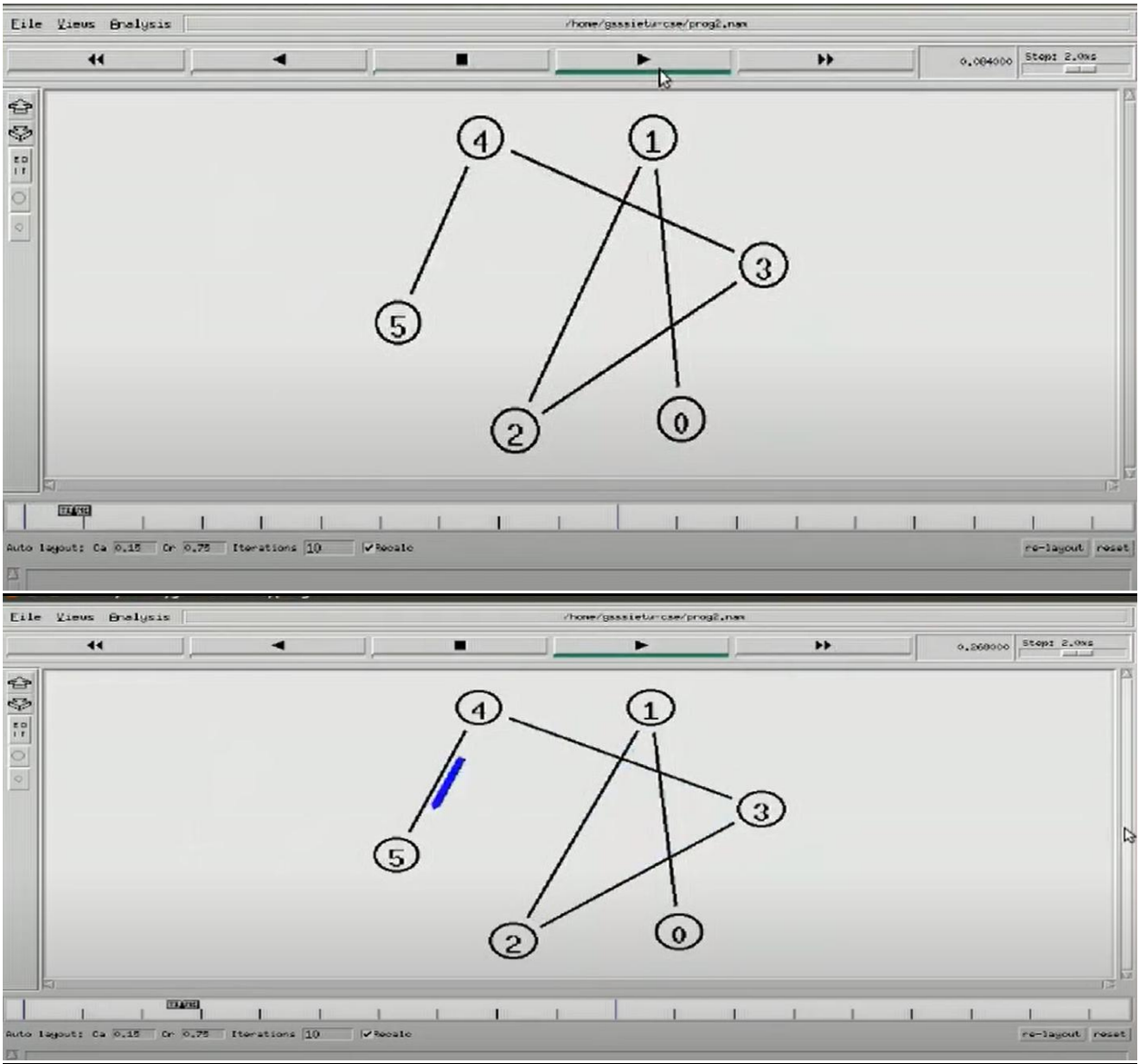
```
#Define the recv function for the class 'Agent/Ping'
Agent/Ping instproc recv {from rtt} {
 $self instvar node_ puts "node [$node_ id] received ping answer from $from with round trip time $rtt ms"
}

#Create two ping agents and attach them to n(0) and n(5)
set p0 [new Agent/Ping]
$p0 set class_ 1
$ns attach-agent $n(0) $p0
set p1 [new Agent/Ping]
$p1 set class_ 1
$ns attach-agent $n(5) $p1
$ns connect $p0 $p1

#Set queue size and monitor the queue
#Queue size is set to 2 to observe the drop in ping packets
$ns queue-limit $n(2) $n(3) 2
$ns duplex-link-op $n(2) $n(3) queuePos 0.5

#Create Congestion
#Generate a Huge CBR traffic between n(2) and n(4)
set tcp0 [new Agent/TCP]
$tcp0 set class_ 2
$ns attach-agent $n(2) $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n(4) $sink0
$ns connect $tcp0 $sink0

#Apply CBR traffic over TCP
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set rate_ 1Mb
$cbr0 attach-agent $tcp0
#Schedule events
$ns at 0.2 "$p0 send"
$ns at 0.4 "$p1 send"
$ns at 0.4 "$cbr0 start"
$ns at 0.8 "$p0 send"
$ns at 1.0 "$p1 send"
$ns at 1.2 "$cbr0 stop"
$ns at 1.4 "$p0 send"
$ns at 1.6 "$p1 send"
```

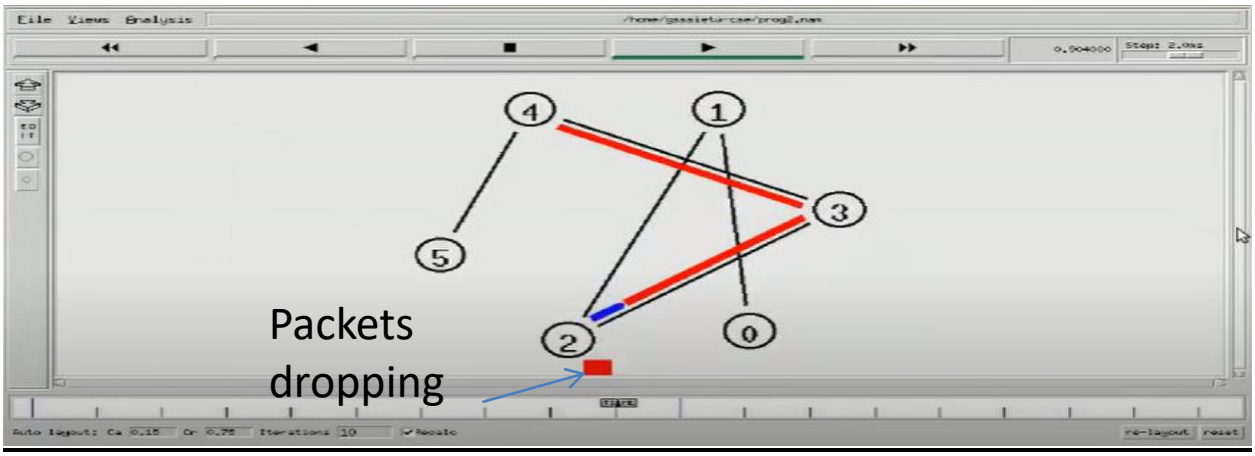
$ns at 1.8 "Finish"

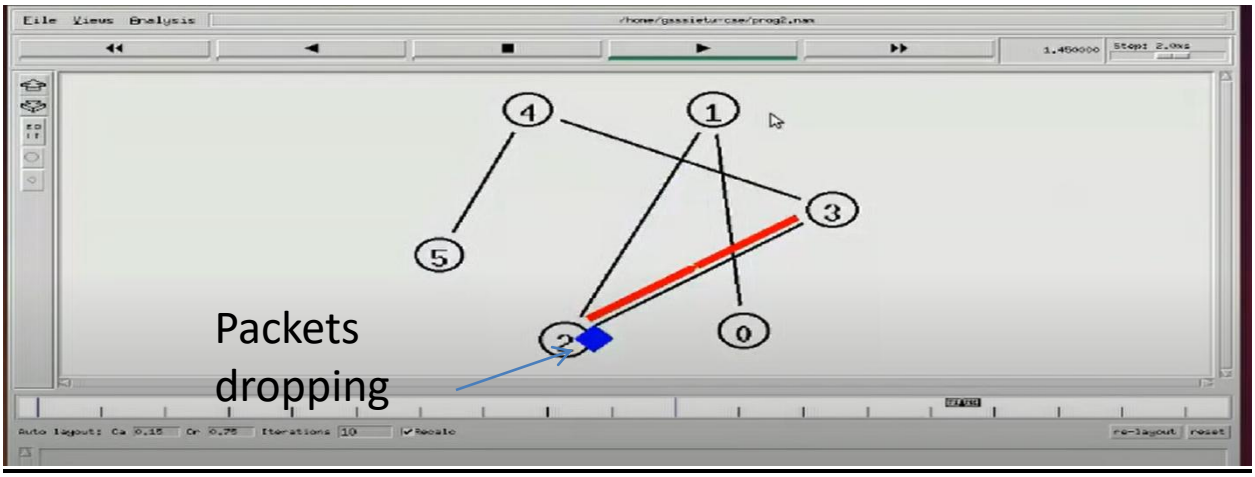
#Run the Simulation
$ns run

Output:

```
Node 0 received ping answer from 5 at round trip
time 151.2 ms
Node 0 received ping answer from 5 at round trip
time 301.4 ms
Node 5 received ping answer from 0 at round trip
time 155.4 ms
Number of ping packets dropped are
3
```




Packets
dropping


Packets
dropping

PROGRAM 3

Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.

```
#Create Simulator

set ns [new Simulator]

#Use colors to differentiate the traffics
$ns color 1 Blue
$ns color 2 Red

#Open trace and NAM trace file
set ntrace [open prog5.tr w]
$ns trace-all $ntrace
set namfile [open prog5.nam w]
$ns namtrace-all $namfile

#Use some flat file to create congestion graph windows
set winFile0 [open WinFile0 w]
set winFile1 [open WinFile1 w]

#Finish Procedure
proc Finish {} {

#Dump all trace data and Close the files
global ns ntrace namfile
$ns flush-trace
close $ntrace
close $namfile

#Execute the NAM animation file
exec nam prog5.nam &

#Plot the Congestion Window graph using xgraph
exec xgraph WinFile0 WinFile1 &
exit 0
}

#Plot Window Procedure
proc PlotWindow {tcpSource file} {
global ns
set time 0.1
set now [$ns now] set cwnd [$tcpSource set cwnd_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "PlotWindow $tcpSource $file"
}
#Create 6 nodes
for {set i 0} {$i<6} {incr i} {set n($i) [$ns node]
}
```

```
#Create duplex links between the nodes
$ns duplex-link $n(0) $n(2) 2Mb 10ms DropTail
$ns duplex-link $n(1) $n(2) 2Mb 10ms DropTail
$ns duplex-link $n(2) $n(3) 0.6Mb 100ms DropTail

#Nodes n(3) , n(4) and n(5) are considered in a LAN
set lan [$ns newLan "$n(3) $n(4) $n(5)" 0.5Mb 40ms LL Queue/DropTail MAC/802_3 Channel]

#Orientation to the nodes
$ns duplex-link-op $n(0) $n(2) orient right-down
$ns duplex-link-op $n(1) $n(2) orient right-up
$ns duplex-link-op $n(2) $n(3) orient right
#Setup queue between n(2) and n(3) and monitor the queue
$ns queue-limit $n(2) $n(3) 20
$ns duplex-link-op $n(2) $n(3) queuePos 0.5

#Set error model on link n(2) to n(3)
set loss_module [new ErrorModel]
$loss_module ranvar [new RandomVariable/Uniform]
$loss_module drop-target [new Agent/Null]
$ns lossmodel $loss_module $n(2) $n(3)

#Set up the TCP connection between n(0) and n(4)
set tcp0 [new Agent/TCP/Newreno]
$tcp0 set fid_ 1 $tcp0 set window_ 8000
$tcp0 set packetSize_ 552
$ns attach-agent $n(0) $tcp0
set sink0 [new Agent/TCPSink/DelAck]
$ns attach-agent $n(4) $sink0
$ns connect $tcp0 $sink0

#Apply FTP Application over TCP
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set type_ FTP

#Set up another TCP connection between n(5) and n(1)
set tcp1 [new Agent/TCP/Newreno]
$tcp1 set fid_ 2
$tcp1 set window_ 8000
$tcp1 set packetSize_ 552
$ns attach-agent $n(5) $tcp1
set sink1 [new Agent/TCPSink/DelAck]
$ns attach-agent $n(1) $sink1
$ns connect $tcp1 $sink1

#Apply FTP application over TCP
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 set type_ FTP
```

```
#Schedule Events
$ns at 0.1 "$ftp0 start"
$ns at 0.1 "PlotWindow $tcp0
$winFile0" $ns at 0.5 "$ftp1 start"
$ns at 0.5 "PlotWindow $tcp1 $winFile1"
$ns at 25.0 "$ftp0 stop"
$ns at 25.1 "$ftp1 stop"
$ns at 25.2 "Finish"

#Run the simulation
$ns run
```

Implement the following in Java:

Program 4

Write a program for error detecting code using CRC-CCITT (16- bits)

```
import java.io.*;

class CRC {
   public static void main(String[] args) throws IOException {

      InputStreamReader isr = new InputStreamReader(System.in);
      BufferedReader br = new BufferedReader(isr);

      int[] message;
      int[] gen;
      int[] app_message;
      int[] rem;
      int[] trans_message;

      int message_bits, gen_bits, total_bits;

      System.out.print("\nEnter number of bits in message: ");
      message_bits = Integer.parseInt(br.readLine());
      message = new int[message_bits];

      System.out.println("Enter message bits:");
      for (int i = 0; i < message_bits; i++) {
         message[i] = Integer.parseInt(br.readLine());
      }

      System.out.print("\nEnter number of bits in generator: ");
      gen_bits = Integer.parseInt(br.readLine());
      gen = new int[gen_bits];

      System.out.println("Enter generator bits:");
      for (int i = 0; i < gen_bits; i++) {
         gen[i] = Integer.parseInt(br.readLine());
      }

      total_bits = message_bits + gen_bits - 1;
      app_message = new int[total_bits];
      rem = new int[total_bits];
      trans_message = new int[total_bits];

      // Copy message into app_message
      for (int i = 0; i < message.length; i++) {
         app_message[i] = message[i];
      }

      System.out.print("\nMessage bits are: ");
      for (int i = 0; i < message_bits; i++) {
```

```
    System.out.print(message[i]);
}

System.out.print("\nGenerator bits are: ");
for (int i = 0; i < gen_bits; i++) {
    System.out.print(gen[i]);
}

System.out.print("\nAppended message is: ");
for (int i = 0; i < app_message.length; i++) {
    System.out.print(app_message[i]);
}

// Copy appended message into remainder
for (int j = 0; j < app_message.length; j++) {
    rem[j] = app_message[j];
}

// Compute CRC
rem = computeCRC(app_message, gen, rem);

// Transmitted message = app_message XOR remainder
for (int i = 0; i < app_message.length; i++) {
    trans_message[i] = (app_message[i] ^ rem[i]);
}

System.out.print("\nTransmitted message from the transmitter is: ");
for (int i = 0; i < trans_message.length; i++) {
    System.out.print(trans_message[i]);
}

// Receiver input
System.out.println("\n\nEnter received message of " + total_bits + " bits at receiver end:");
for (int i = 0; i < trans_message.length; i++) {
    trans_message[i] = Integer.parseInt(br.readLine());
}

System.out.print("\nReceived message is: ");
for (int i = 0; i < trans_message.length; i++) {
    System.out.print(trans_message[i]);
}

// Copy received message into remainder
for (int j = 0; j < trans_message.length; j++) {
    rem[j] = trans_message[j];
}

rem = computeCRC(trans_message, gen, rem);

boolean error = false;
for (int i = 0; i < rem.length; i++) {
```

```
            if (rem[i] != 0) {
                System.out.println("\n\nThere is an Error in the received message!!!");
                error = true;
                break;
            }
        }

        if (!error) {
            System.out.println("\n\nThere is No Error in the received message!!!");
        }
    }

    // Function to compute CRC remainder
    static int[] computeCRC(int app_message[], int gen[], int rem[]) {
        int current = 0;

        while (true) {
            for (int i = 0; i < gen.length; i++) {
                rem[current + i] = (rem[current + i] ^ gen[i]);
            }

            while (rem[current] == 0 && current != rem.length - 1) {
                current++;
            }

            if ((rem.length - current) < gen.length) {
                break;
            }
        }

        return rem;
    }
}
```

Output:

```
Enter number of bits in message: 5
Enter message bits:
1
0
1
1
0



Enter number of bits in generator: 3
Enter generator bits:
1
```

0
1

Message bits are: 10110
Generator bits are: 101
Appended message is: 1011000
Transmitted message from the transmitter is: 1011010

Enter received message of 7 bits at receiver end:
1
0
1
1
0
1
1

Received message is: 1011011

There is an Error in the received message!!!

Program 5

Develop a program to implement a sliding window protocol in the data link layer.

//Sender Side

```
import java.util.LinkedList;
import java.util.Queue;
public class SlidingSender {
	private int windowSize;
	private Queue<Integer> window;
	private int nextSeqNum;
	private int ackNum;


	public SlidingSender(int windowSize) {
	this.windowSize=windowSize;
	this.window=new LinkedList<>();
	this.nextSeqNum=0;
	this.ackNum = 0;
	}
	public void sendPacket() {
		while(nextSeqNum <ackNum + windowSize) {
			System.out.println("Sending Packet:" + nextSeqNum);
			window.add(nextSeqNum);
			nextSeqNum++;
			try {
				Thread.sleep(500);
			   }	catch(InterruptedException e) {
				e.printStackTrace();
			}
			}
		}
		public void receiveAck(int ack) {
			if (ack>ackNum && ack <nextSeqNum) {
				System.out.println("Received ACK for packet:"+ack);
				ackNum=ack;

				//Remove acknowledged packets if possible
				while(!window.isEmpty() && window.peek() <=ackNum) {
					window.poll();
				}
				sendPacket();
			} else {
				System.out.println("Invalid ACK:"+ack);
			}
		}

				public static void main(String[] args) {
					int windowSize=3;
					SlidingSender sender=new SlidingSender(windowSize);
```

```
                        // Start sending packets
                        sender.sendPacket();

                        // Simulate receiving acknowledgments
                        SlidingReceiver receiver=new SlidingReceiver(sender);
                        receiver.stimulateReceiver();


                }
        }
```

// Receiver Side

```
package CNLab;
//package Swathi;
import java.util.Scanner;

public class SlidingReceiver {
private SlidingSender sender;

                public SlidingReceiver(SlidingSender sender) {
                        this.sender=sender;
                        }

                public void stimulateReceiver() {
                        Scanner scanner=new Scanner(System.in);
                        while (true) {
                                System.out.print("Enter ACK number(or-1 to exit):");
                                int ack=scanner.nextInt();
                                if (ack==-1)
                                { System.out.println("Exiting.....");
                                break;
                                }
                                sender.receiveAck(ack);
                        }
                        scanner.close();
                        }

                }
```

Output:

```
Sending Packet:0
Sending Packet:1
Sending Packet:2
Enter ACK number(or-1 to exit):0
Invalid ACK:0
Enter ACK number(or-1 to exit):1
Received ACK for packet:1
Sending Packet:3
Enter ACK number(or-1 to exit):3
Received ACK for packet:3
Sending Packet:4
Sending Packet:5
Enter ACK number(or-1 to exit):-1
Exiting.....
```

Program 6

Develop a program to find the shortest path between vertices using the Bellman-Ford and path vector routing algorithm.

```
import java.util.Scanner;

public class BellMan_Ford {
    private int D[];
    private int num_ver;
    public static final int MAX_VALUE = 999;

    public BellMan_Ford(int num_ver) {
        this.num_ver = num_ver;
        D = new int[num_ver + 1];
    }

    public void BellmanFordEvaluation(int source, int A[][]) {
        for (int node = 1; node <= num_ver; node++) {
            D[node] = MAX_VALUE;
        }
        D[source] = 0;

        for (int node = 1; node <= num_ver - 1; node++) {
            for (int sn = 1; sn <= num_ver; sn++) {
                for (int dn = 1; dn <= num_ver; dn++) {
                    if (A[sn][dn] != MAX_VALUE) {
                        if (D[dn] > D[sn] + A[sn][dn]) {
                            D[dn] = D[sn] + A[sn][dn];
                        }
                    }
                }
            }
        }

        // Check for negative-weight cycles
        for (int sn = 1; sn <= num_ver; sn++) {
            for (int dn = 1; dn <= num_ver; dn++) {
                if (A[sn][dn] != MAX_VALUE) {
                    if (D[dn] > D[sn] + A[sn][dn]) {
                        System.out.println("The Graph contains negative edge cycle");
                    }
                }
            }
        }
```

```
        // Print shortest distances
        for (int vertex = 1; vertex <= num_ver; vertex++) {
            System.out.println("Distance of source " + source + " to " + vertex + " is " + D[vertex]);
        }
    }

    public static void main(String[] args) {
        int num_ver = 0;
        int source;

        Scanner sca = new Scanner(System.in);

        System.out.println("Enter the number of vertices:");
        num_ver = sca.nextInt();

        int A[][] = new int[num_ver + 1][num_ver + 1];
        System.out.println("Enter the adjacency matrix:");

        for (int sn = 1; sn <= num_ver; sn++) {
            for (int dn = 1; dn <= num_ver; dn++) {
                A[sn][dn] = sca.nextInt();

                if (sn == dn) {
                    A[sn][dn] = 0;
                    continue;
                }

                if (A[sn][dn] == 0) {
                    A[sn][dn] = MAX_VALUE;
                }
            }
        }

        System.out.println("Enter the source vertex:");
        source = sca.nextInt();

        BellMan_Ford b = new BellMan_Ford(num_ver);
        b.BellmanFordEvaluation(source, A);

        sca.close();
    }
}
```

Output:

```
Enter the number of vertices:
5
Enter the adjacency matrix:

0 6 5 0 0
0 0 0 -1 0
0 -2 0 4 3
0 0 0 0 3
0 0 0 0 0
Enter the source vertex:
1
distance of source 1 to 1 is 0
distance of source 1 to 2 is 3
distance of source 1 to 3 is 5
distance of source 1 to 4 is 2
distance of source 1 to 5 is 5
```

Program 7

Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present

Server Side:

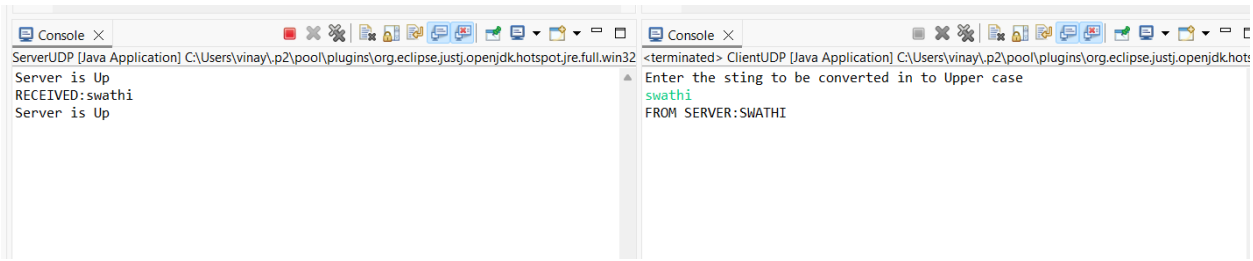
```
package CNLab;

import java.net.*;

class ServerUDP {
   public static void main(String args[]) throws Exception {
      DatagramSocket serverSocket = new DatagramSocket(9876);
      byte[] receiveData = new byte[1024];
      byte[] sendData = new byte[1024];

      while (true) {
         System.out.println("Server is Up");

         // Receive data from client
         DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
         serverSocket.receive(receivePacket);

         // Convert received data to string
         String sentence = new String(receivePacket.getData()).trim(); // `.trim()` to remove padding
         System.out.println("RECEIVED: " + sentence);

         // Get client IP and port
         InetAddress IPAddress = receivePacket.getAddress();
         int port = receivePacket.getPort();

         // Convert to uppercase
         String capitalizedSentence = sentence.toUpperCase();
         sendData = capitalizedSentence.getBytes();

         // Send back to client
         DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, port);
         serverSocket.send(sendPacket);
      }
   }
}
```

Client Side:

```
package CNLab;

import java.io.*;
import java.net.*;

class ClientUDP {
```

```
    public static void main(String[] args) throws Exception {
        // Read input from user
        BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));

        // Create UDP socket
        DatagramSocket clientSocket = new DatagramSocket();

        // Get server IP address
        InetAddress IPAddress = InetAddress.getByName("localhost");

        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];

        System.out.println("Enter the string to be converted into Upper Case:");
        String sentence = inFromUser.readLine();

        // Convert string to bytes
        sendData = sentence.getBytes();

        // Create packet to send to server
        DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, 9876);
        clientSocket.send(sendPacket);

        // Receive response from server
        DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
        clientSocket.receive(receivePacket);

        // Convert received data to string and trim
        String modifiedSentence = new String(receivePacket.getData()).trim();
        System.out.println("FROM SERVER: " + modifiedSentence);

        // Close the socket
        clientSocket.close();
    }
}
```

Output



Program 8

Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present

Server Side Program:

```
package CN;

import java.net.*;
import java.io.*;

public class TCPServer {
  public static void main(String[] args) throws Exception {
      // Create server socket listening on port 4000
      ServerSocket sersock = new ServerSocket(4000);
      System.out.println("Server ready for connection");

      // Accept incoming client connection
      Socket sock = sersock.accept();
      System.out.println("Connection is successful and waiting for file request...");

      // Receive the filename from the client
      InputStream istream = sock.getInputStream();
      BufferedReader fileRead = new BufferedReader(new InputStreamReader(istream));
      String fname = fileRead.readLine();

      // Read content from the requested file
      BufferedReader ContentRead = new BufferedReader(new FileReader(fname));
      OutputStream ostream = sock.getOutputStream();
      PrintWriter pwrite = new PrintWriter(ostream, true);

      // Send file content to client
      String str;
      while ((str = ContentRead.readLine()) != null) {
          pwrite.println(str);
      }

      // Close all resources
      sock.close();
      sersock.close();
      pwrite.close();
      fileRead.close();
      ContentRead.close();
  }
}
```

Client Side Program:

```
package CNLab;

import java.net.*;
import java.io.*;

public class TCPClient {
  public static void main(String[] args) throws Exception {
      // Connect to the server at localhost on port 4000
      Socket sock = new Socket("127.0.0.1", 4000);

      System.out.println("Enter the filename:");

      // Read filename from the keyboard
      BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));
      String fname = keyRead.readLine();

      // Send filename to the server
      OutputStream ostream = sock.getOutputStream();
      PrintWriter pwrite = new PrintWriter(ostream, true);
      pwrite.println(fname);

      // Receive file content from server
      InputStream istream = sock.getInputStream();
      BufferedReader socketRead = new BufferedReader(new InputStreamReader(istream));

      String str;
      while ((str = socketRead.readLine()) != null) {
          System.out.println(str);
      }

      // Close all resources
      // sock.close(); // Uncomment this if you want to explicitly close the socket
      pwrite.close();
      socketRead.close();
      keyRead.close();
  }
}
```

Output

1. Run Server Side Program 1st.

Server ready for connection

2. Run Client Side Program Next.

Server ready for connection
Connection is successful and waiting for file request...

3. Sample.txt file should be present under the project, i.e Server side.

4. Enter the filename in the client Side; the file content should be displayed at client side.

Enter the filename:
Sample.txt
Hello good morning all

Program 9

Develop a program for a simple RSA algorithm to encrypt and decrypt the data.

```
import java.io.DataInputStream;
import java.io.IOException;
import java.math.BigInteger;
import java.util.Random;

public class RSA {
    private BigInteger p, q, N, phi, e, d;
    private int bitlength = 1024;
    private Random r;

    // Constructor to generate keys
    public RSA() {
        r = new Random();
        p = BigInteger.probablePrime(bitlength, r);
        q = BigInteger.probablePrime(bitlength, r);
        System.out.println("Prime number p is: " + p);
        System.out.println("Prime number q is: " + q);

        N = p.multiply(q);
        phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
        e = BigInteger.probablePrime(bitlength / 2, r);

        // Ensure e is coprime to phi
        while (phi.gcd(e).compareTo(BigInteger.ONE) != 0 && e.compareTo(phi) < 0) {
            e = e.add(BigInteger.ONE);
        }
```

```
        System.out.println("Public key (e): " + e);
        d = e.modInverse(phi);
        System.out.println("Private key (d): " + d);
    }

    // Constructor with provided keys
    public RSA(BigInteger e, BigInteger d, BigInteger N) {
        this.e = e;
        this.d = d;
        this.N = N;
    }

    // Main method
    public static void main(String[] args) throws IOException {
        RSA rsa = new RSA();
        DataInputStream in = new DataInputStream(System.in);

        System.out.print("Enter the plain text: ");
        String testString = in.readLine();

        System.out.println("Encrypting string: " + testString);
        System.out.println("String in bytes: " + bytesToString(testString.getBytes()));

        byte[] encrypted = rsa.encrypt(testString.getBytes());
        System.out.println("Encrypted string in bytes: " + bytesToString(encrypted));

        byte[] decrypted = rsa.decrypt(encrypted);
        System.out.println("Decrypted Bytes: " + bytesToString(decrypted));
        System.out.println("Decrypted string: " + new String(decrypted));
    }
```

```
    // Convert byte array to string for display
    private static String bytesToString(byte[] bytes) {
        StringBuilder sb = new StringBuilder();
        for (byte b : bytes) {
            sb.append(Byte.toString(b));
        }
        return sb.toString();
    }

    // Encrypt method
    public byte[] encrypt(byte[] message) {
        return (new BigInteger(message)).modPow(e, N).toByteArray();
    }

    // Decrypt method
    public byte[] decrypt(byte[] message) {
        return (new BigInteger(message)).modPow(d, N).toByteArray();
    }
}
```

Output:

Prime number p is:

166254061285867130815508011112720181739600563922395697584266981390316074049678523324114
711333805214444528778358836548911181902609927503702206780674336747499171757650461269352
838390975260414343677065191885887546366358822777545100700226046575735372748112888042313
834059774021161631174766689644190009339872437189

Prime number q is:

119040734405635099787180920862754870640432126637422910484101882332343360571147458293380
146586842339174825049282717331608823360919505662714916519123878673849237415233571519604
015475772744564043969084408004454535256841848849583234534532035818541371416056068185124
976688259898119714455168640484413970633706291237

Public key (e):

734494768441806975813691307046228943092958634187960907743800792168124423218749465524637
748117373908250140933357722136341750833676255312102029565995600910 9

Private key (d):

563513174223105709515571108777947218943294091828350340758706236923656192906564566565512
973185253764998563263483231908163823389868508824329043015508041041573880182120163749215
539183531316756712370811535771894804574717074503144410624825475647741397587881727425671
782140397623541680132047125378598486540825284700582128975689397185094063459688894223023
937565285503549710828212049450825419464987300094789673795384847003837818899855321320080
526546475760762649105736974813351696636917775034208213614756631916467677414113482676122
524593507058623244698214316590172088073252568317786029461603848107983068763135807924545
5514637

Enter the plain text:

hello how are you

Encrypting string:

hello how are you

String in bytes: 104101108108111321041111193297114101321211111117

Encrypted string in bytes:

0-109-12-994-845562-42-49127368-11720-10-67121-9064-123-70-8388-66433-1587108-103-109-4799-5110725-24-83-36-46-37-68-717396-102-25-78-4733611112-5211-107-70-9914-4182-5-10-77-94115-112-218239476-61-2710485-35-70-28-12658-3045-104-44-521241054698-105-57-6480-11-107-61-365225-32-56-26-391161192132758798886101686 7-2788-114114-12611344-59-24105-64-1134387-108101-71124-86-27-6729-78881416108-317698-72-72-222951-90186795879819-797110554-63578-7981-122-12090-5629111-42107117832150-103-14-58-94-740394970-27-1132312211-10641-111-23-5510279-7691162914-115-8619876-75-62-199-61-101117-8811332231-11043-451211037515239829-19114-8789186-6919-28-2553-70-9427-7981-4212-84-83

Decrypted Bytes: 104101108108111321041111193297114101321211111117

Decrypted string: hello how are you

Program 10

Develop a program for congestion control using a leaky bucket algorithm.

```
import java.util.Scanner;

public class Leaky {
    public static void main(String[] args) {
        int i;
        int a[] = new int[20];
        int buck_rem = 0, buck_cap = 4, rate = 3, sent, recv;

        Scanner in = new Scanner(System.in);

        System.out.println("Enter the number of packets:");
        int n = in.nextInt();

        System.out.println("Enter the packet sizes:");
        for (i = 1; i <= n; i++) {
            a[i] = in.nextInt();
        }

        System.out.println("Clock \t Packet Size \t Accept \t Sent \t Remaining");

        for (i = 1; i <= n; i++) {
            if (a[i] != 0) {
                if (buck_rem + a[i] > buck_cap) {
                    recv = -1; // packet dropped
                } else {
                    recv = a[i];
                    buck_rem += a[i];
```

```
            }
        } else {
            recv = 0;
        }

        if (buck_rem != 0) {
            if (buck_rem < rate) {
                sent = buck_rem;
                buck_rem = 0;
            } else {
                sent = rate;
                buck_rem -= rate;
            }
        } else {
            sent = 0;
        }
        if (recv == -1) {
            System.out.println(i + "\t\t" + a[i] + "\t dropped \t" + sent + "\t" + buck_rem);
        } else {
            System.out.println(i + "\t\t" + a[i] + "\t\t" + recv + "\t\t" + sent + "\t" + buck_rem);
        }
    }
    in.close();
  }
}
```

Output

Enter the number of packets:

5

Enter the packet sizes:

2

3

5

4

7

Clock	Packet Size	Accept	Sent	Remaining
1	2	2	2	0
2	3	3	3	0
3	5	dropped	0	0
4	4	4	3	1
5	7	dropped	1	0

Viva Questions

1. Explain the functions of OSI layers ?
2. Differentiate between TCP/IP Layers and OSI Layers
3. Why header is required?
4. What is the use of adding header and trailer to frames?
5. What is encapsulation?
6. Why fragmentation requires?
7. What is MTU?
8. Which layer imposes MTU?
9. Differentiate between flow control and congestion control.
10. Differentiate between Point-to-Point Connection and End-to-End connections.
11. What are protocols running in different layers?
12. What is Protocol Stack?
13. Differentiate between TCP and UDP.
14. Differentiate between Connectionless and connection oriented connection.
15. Why frame sorting is required?
16. What is meant by subnet?
17. What is meant by Gateway?
18. What is an IP address?
19. What is MAC address?
20. Why IP address is required when we have MAC address?
21. What is meant by port?
22. What are ephemerical port number and well known port numbers?
23. What is a socket?
24. What are the parameters of socket()?
25. Describe bind(), listen(), accept(),connect(), send() and recv().
26. What are system calls? Mention few of them.
27. What is IPC? Name three techniques.
28. Explain mkfifo(), open(), close() with parameters.
29. What is meant by file descriptor?
30. What is meant by traffic shaping?
31. How do you classify congestion control algorithms?
32. Differentiate between Leaky bucket and Token bucket.
33. How do you implement Leaky bucket?
34. How do you generate bursty traffic?
35. What is the polynomial used in CRC-CCITT?
36. What are the other error detection algorithms?
37. What is difference between CRC and Hamming code?
38. Why Hamming code is called 7,4 code?
39. What is odd parity and even parity?
40. What is meant by syndrome?
41. What is generator matrix?
42. What are Routing algorithms?
43. How do you classify routing algorithms? Give examples for each.
44. What are drawbacks in distance vector algorithm?
45. How routers update distances to each of its neighbor?
46. How do you overcome count to infinity problem?
47. What is cryptography?

48. How do you classify cryptographic algorithms?
49. What is public key?
50. What is private key?
51. What are key, ciphertext and plaintext?
52. What is simulation?
53. What are advantages of simulation?
54. Differentiate between Simulation and Emulation.
55. What is meant by router?
56. What is meant by bridge?
57. What is meant by switch?
58. What is meant by hub?
59. Differentiate between route, bridge, switch and hub.
60. What is ping and telnet?
61. What is FTP?
62. What is BER?
63. What is meant by congestion window?
64. What is BSS?
65. What is incoming throughput and outgoing throughput?
66. What is collision?
67. How do you generate multiple traffics across different sender-receiver pairs?
68. How do you setup Ethernet LAN?
69. What is meant by mobile host?
70. Name few other Network simulators
71. Differentiate between logical and physical address.
72. Which address gets affected if a system moves from one place to another place?
73. What is ICMP? What are uses of ICMP? Name few.
74. Which layer implements security for data?
75. What is flow control and error control?
76. What are the protocols used for error control in data link layer.
77. What is the difference between?
    a. Transferring frames from source to destination.
    b. Transferring packets from source to destination.
    c. Transferring segments from source to destination.
78. Why the message is divided into segments?
79. How do we define End to End communication messages using 5 tuples?
80. What is Routing? What are the different routing protocols used in N/W layer?
81. What is meant by gateway?
82. What are the different N/W topologies?
83. What is the topology used in your lab?
84. What are the three address used to identify a system? Name them? What is their importance?
85. What is the need of MAC protocols? Give examples.
86. Difference between CSMA/CA and CSMA/CD.
87. Why CSMA/CD is not used in WLANS?
88. How CSMA/CA is adopted in WLANS?
89. What is simulation.Why is it necessary.
90. Mentions some simulator tools.